UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Threat detection through data analysis using machine learning algorithms

Fernando Martín Robles

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Threat detection through data analysis using machine learning algorithms

Autor: Fernando Martín Robles
Tutor: Fernando Díez Rubio

julio 2020

Fernando Martín Robles
Threat detection through data analysis using machine learning algorithms

Fernando Martín Robles
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mis compañeros de trabajo por darme la idea, a mi padre y mi madre de los que aprendí a no rendirme, y a mi tutor, por su paciencia y su apoyo durante estos meses tan difíciles

Es una locura odiar a todas las rosas porque una te pinchó.
Renunciar a todos tus sueños porque uno de ellos no se realizó.

El Principito

Prefacio

En un mundo cada vez más informatizado la necesidad de defenderse de las amenazas digitales toma un papel indispensable para prácticamente cualquier empresa. Las técnicas utilizadas en los ataques informáticos son cada vez más sofisticadas y cambian más rápido y por ello hay que encontrar maneras más eficaces y rápidas para enfrentarse a los atacantes, esto incluye la detección, categorización y la posterior respuesta al ataque.

La detección es llevada acabo mediante reglas también denominadas alertas o casos de uso en un **SIEM** (Security Information and Event Management). Estas alertas toman información de servidores, sistemas operativos, proxies, firewalls... De tal manera que categorizan el ataque y dan un contexto de trabajo para los equipos **SOC** (Security Operations Center) y **Response**.

Por tanto es importante **reducir el tiempo de este proceso**, en particular, en las etapas de generación de una alerta y del posterior análisis de la amenaza del equipo SOC para generar la primera respuesta.

Debido a los altos niveles de información que se tienen que gestionar a día de hoy en ciberseguridad, es difícil para los analistas realizar un análisis completo y eficiente en un periodo de tiempo corto, por ello se propone la utilización de **ciencia de datos** y **aprendizaje automático** para reducir tiempo y costes.

Fernando Martín Robles

Resumen

En este **TFG** (Trabajo de Fin de Grado), se busca simular el trabajo de un **SIEM** y automatizar la primera respuesta ante un ciberataque utilizando modelos de machine learning para predecir y generar una respuesta automática.

Para ello primero se ha realizado un estudio sobre el dataset **KDD Cup 1999**, el cual recoge información sobre distintos ciberataques organizados bajo **4 categorías principales: dos, probe, r2l y u2r.**

Se han definido y analizado **18 ciberataques distintos** y se informa sobre como proceder cuando se recibe cualquiera de estos ataques, o lo que es lo mismo, un **playbook**.

Tras ello se analiza la posibilidad de crear predicciones utilizando **modelos**. Se realiza un estudio sobre cada uno de los modelos y su tasa de éxito para, finalmente, crear un **modelo híbrido** utilizando los 3 mejores modelos del estudio.

Palabras clave

ciberseguridad, aprendizaje automático, detección de amenazas, ciberataques, ciencia de datos, python

Abstract

In this TFG, we try to simulate a **SIEM** and try to find a way to automate the first awnser after reciving a cyberattack using machine learning models to predict and generate an automatic awnser.

To acomplish this, first i realized a study of the dataset **KDD Cup 1999** which contains information about different cybearttacks under **4 major categories: dos, probe, r2l y u2r.**

I have defined and analyzed **18 different cybearttacks** and report how to proceed when you receive any of those cyberattacks, which is called **playbook**.

After that, i analyzed the possibility of creating predictions using **models**. I made a study of the models and it's success rate to, finally, create an **hybrid model** by using the top 3 best performing models

Keywords

Cybersecurity, machine learning, threat detection, cyberattacks, data science, python

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1

Motivación e Introducción

1.1. Motivación

El ámbito de este TFG es intentar mejorar la velocidad para emitir una primera respuesta ante un ciberataque uniendo los mundos de **la ciberseguridad y el aprendizaje automático**.

Con ello lo que se quiere conseguir es proporcionar un nivel de respuesta muy rápido para ataques comunes para que los analistas de ciberseguridad pertenecientes al **SOC**, Security Operations Center, dispongan de mayor tiempo para incidentes de mayor criticidad así como intentar parar el ataque en una etapa temprana de la **Cyber Kill Chain**.

La **Cyber Kill Chain** es una serie de etapas en las que se pueden dividir los ataques informáticos.

Figura 1.1: Etapas de la kill chain de ciberseguridad. Foto de INCIBE (Instituto nacional de ciberseguridad) [1]

Fase 1 ***Reconocimiento.*** *En esta fase el atacante recopila información sobre el objetivo. Los ataques de tipo probing, de los que se hablará más adelante, pertenecen a esta sección.*

Fase 2 ***Presentación.*** *El atacante prepara lo necesario para el ataque y crea los mecanismos necesarios para el mismo.*

Fase 3 ***Distribución.*** *En esta etapa se produce la transmisión y comienza el ataque.*

Fase 4 ***Explotación.*** *En esta fase se compromete al obejtivo del ataque.*

Fase 5 ***Instalación.*** *El atacante instala malware en la máquina objetivo.*

Fase 6 ***Comando y Control.*** *El atacante toma el control sobre la máquina objetivo obteniendo acceso a todas sus características e información.*

Fase 7 ***Acción sobre los objetivos.*** *El atacante se hace con los datos que desea e intenta moverse lateralmente para aumentar su influencia sobre la red objetivo.*

1.2. Introducción

Basándonos en las motivaciones anteriores, este TFG puede servir como punto de partida para unir la ciberseguridad y el aprendizaje automático para llevar la detección y respuesta de ciberataques a un nivel superior.

En el **capitulo 2**, se da un contexto más profundo del trabajo, así como una explicación de la estrategia seguida y una introducción al dataset utilizado en el trabajo, que se corresponde con el dataset de la **KDD Cup 1999** que contiene 5 millones de eventos relacionados con ciberataques.

En el **capítulo 3** se aborda el problema que se ha enfrentado en este TFG, elaborar un estudio y la predicción de los ciberataques del dataset.

El **capítulo 4**, muestra los resultados del estudio del dataset, esto incluye una **explicación de cada uno de los campos del dataset** así como un **análisis de cada uno de los ciberataques** junto a su definición y sus contramedidas. También se pueden encontrar las definiciones y características de los **modelos** utilizados para predecir.

Los **resultados del experimento** se encuentran en el **capítulo 5** entre los que se pueden encontrar cuales son los 3 modelos con mayor porcentaje de aciertos en su predicción y la información referente al **modelo híbrido** que se propone.

Finalmente, el trabajo contiene **las conclusiones**, ideas para el futuro y **la bibliografía**.

2

Estado del arte

2.1. Contexto del trabajo

A este tipo de software que automatiza la gestión y respuesta de eventos de ciberseguridad se les denomina **SOAR** (Security Orchestration, Automation, and Response).

El concepto de **SOAR** es relativamente nuevo y parte del concepto de ICOPs (Integrated Cybersecurity Orchestration Platforms), la función de los ICOPs era coleccionar datos de distintas aplicaciones de seguridad para correlarlos y enriquecerlos. [2]

Como ejemplo de SOAR en la actualidad se puede tomar **Splunk Phantom**. [3] **Splunk** es un **SIEM** (Security Information and Event Management). La principal funcionalidad de un **SIEM** es reunir la información importante dentro de la red de una empresa, estandarizar datos y permitir la creación de reglas para detectar ciberataques, los cuales se denominan casos de uso. **Phantom** añade la capacidad de implementar **playbooks automatizados** para cada caso de uso de tal manera, que agiliza la primera respuesta ante un ataque permitiendo bloquear una url, buscar un archivo dañino dentro de tu red o dejar un activo infectado en cuarentena entre otras características.

Este TFG se centra en desarrollar un estudio sobre el dataset **KDD Cup 1999**, definiendo los tipos de ciberataques que se encuentran en él, esta parte del trabajo además de ser la base para entender nuestros datos, sirve para aprender sobre los ciberataques que se mencionan. Después se utilizan **modelos para predecir futuros ataques** de los mismos tipos a partir de logs nuevos que se vayan introduciendo. Este es un funcionamiento similar a un **SIEM** sustituyendo los casos de uso por las predicciones.

Al no disponer de una red sobre la que tomar decisiones, el programa tan solo maneja una tabla con ataques y respuestas, este funcionamiento sería similar a un SOAR pero sin automatizar.

2.2. Estrategia de actuación

En primer lugar se ha revisado el **estado del dataset**, y se ha condicionado a las circunstancias, debido a las resitrcciones de cómputo en las que se ha desarrolado el trabajo, se ha tomado la decisión de reducir el número de eventos con los que trabajo a **50.000**.

El segundo paso fue comprender el significado de cada campo del dataset (cada campo se corresponde con una columna).

Tras ello he elegido **Python** para desarrollar el scripting del proyecto, utilizando principalmente las librerías **Pandas y Sklearn**. Tras conocer el funcionamiento de las funciones de dichas librerías decidí **modificar los campos del dataset con valores no númericos** y transformarlos en numéricos ya que estos si podían ser utilizados a la hora de realizar las predicciones.

El cuarto paso es la codificación en **Python** para estudiar cada uno de los ataques, esto incluye **histogramas y gráficos de caja** para cada uno de los **campos**.

La siguiente acción que ha sido llevada acabo es el **código** para usar los **modelos predictivos** en el dataset junto a sus respectivos resultados.

El sexto y último paso es codificar el **modelo híbrido** utilizando los tres mejores modelos en base a su tasa de éxito.

2.3. Introducción al dataset

El dataset utilizado proviene de la **KDD Cup 1999**. Las siglas **KDD** se corresponden a "Knowledge Discovery and Data Mining Tools Competition", y es una competición de la universidad de California, Irvine en Estados Unidos.

Es un dataset ampliamente utilizado en multitud de proyectos de minería de datos, aprendizaje automático, aprendizaje profundo y ciberseguridad en Google Scholar. [4] [5]

También se encuentra en plataformas importantes como Kaggle, que proporciona datasets para aprender y trabajar la ciencia de datos.

3

Descripción del problema - Elección del dataset

3.1. Descripción del problema

A la vista de la información de los capitulos anteriores, el problema que se plantea es el siguiente. ¿Se puede emular el funcionamiento de un **SIEM** o un **SOAR** tomando como base el dataset de la **KDD Cup 1999**?

Veremos que el comportamiento de un **SIEM/SOAR** se puede conseguir con un pequeño script de **Python** gracias al aprendizaje automático, pero antes de eso, primero, hay que conocer el contenido del **dataset KDD Cup 1999**. [6]

3.2. Elección del dataset

El **dataset de la KDD Cup 1999** contiene **5 millones** de eventos de ciberataques junto a logs normales, y a pesar de que el mundo de la ciberseguridad está en constante cambio, su estructura bien definida y los ejemplos de ciberataques que se incluyen han hecho que sea uno de los más populares.

3.2.1. Campos del dataset

Entrando más en detalle, el dataset contiene **41 columnas o campos**, de **tres tipos de datos distintos**: Cadena de caracteres, numéricos y booleano (o bien valen 1 o 0).

La siguiente tabla contiene la información de los campos del dataset

Nota: Aquellos campos con (*) se refieren a conexiones a través de un mismo servicio.

Campo	Descripción	Tipo
duration	tiempo (número de segundos) de la conexión	numérico
protocol type	tipo del protocolo, e.j. tcp, udp, etc.	cad. caracteres
service	servicio utlizado, e.j. http, telnet, etc.	cad. caracteres
src bytes	número de bytes enviados desde src hasta dest	numérico
dst bytes	número de bytes enviados desde dest hasta src	numérico
flag	indica estados de la conexión	cad. caracteres
land	1 si la conexión es por los mismo puertos, 0 en caso de ser distintos	booleano
wrong fragment	número de fragmentos erróneos	numérico
urgent	número de paquetes urgentes	numérico
hot	número de indicadores hot	numérico
num failed logins	número de errores de autenticación	numérico
logged in	1 si usuario loggeado, 0 en otro caso	booleano
num compromised	número de condiciones comprometidas	numérico
root shell	1 si tiene permisos de root, 0 en otro caso	booleano
su attempted	1 si se ha intentado usar el comando "su", 0 en otro caso	booleano
num root	número de accesos como root	numérico
num file creations	número de operaciones con ficheros	numérico
num shells	numero de shells operativas	numérico
num access files	número de operaciones relacionadas con acceso a ficheros	numérico
num outbound cmds	número de comandos externos durante una sesión ftp	numérico
is hot login	1 si el login pertenece a la hot list, 0 en otro caso	booleano
is guest login	1 si el login es de tipo guest, 0 en otro caso	booleano
count	número de conexiones al mismo host de esta conexión	numérico
serror rate	% de conexiones que tienen errores SYN	numérico
rerror rat	% de conexiones que tienen errores REJ	numérico
same srv rate	% de conexiones al mismo servicio	numérico
diff srv rate	% de conexiones a diferentes servicios	numérico
srv count	número de conexiones *	numérico
srv serror rate	% de conexiones que tienen errores SYN *	numérico
srv rerror rate	% de conexiones que tienen errores REJ *	numérico
srv diff host rate	% de conexiones a distintos hosts *	numérico

Tabla 3.1: Campos del dataset I [7]

Campo	Descripción	Tipo
dst host count	número de conexiones del host destino	numérico
dst host srv count	número de conexiones del host destino *	numérico
dst host same srv rate	% de uso del mismo servicio del host destino	numérico
dst host diff srv rate	% de uso de distintos servicios del host destino	numérico
dst host same src port rate	% de uso de un mismo puerto origen del host destino	numérico
dst host srv diff host rate	% de uso de servicios en función de un host	numérico
dst host serror rate	% de conexiones con errores SYN	numérico
dst host srv serror rate	% de conexiones con errores SYN *	numérico
dst host rerror rate	% de conexiones con errores REJ	numérico
dst host srv rerror rate	% de conexiones con errores REJ *	numérico

Tabla 3.2: Campos del dataset II [7]

3.2.2. Cambios aplicados al dataset

Para poder realizar el estudio correctamente se ha tenido que tomar una muestra de unos **50000 eventos** para evitar una sobrecarga de la máquina dónde se ha llevado a cabo la experimentación.

Se modificaron los valores de los campos string por campos numéricos, ya que las funciones de los modelos solo toman valores numéricos.

Los **3 campos que se modificaron** por ser de tipo cadena de caracteres y cambiaron a numéricos fueron: **protocol type, service y flag**. Los cambios se reflejan en las siguientes tablas.

protocol type	Number	flag	Number
icmp	0	SF	0
tcp	1	S1	1
udp	2	REJ	2
		S2	3
		S0	4
		S3	5
		RSTO	6
		RSTR	7
		OTH	8
		SH	9
		RSTO4	10

Tabla 3.3: Protocol type y flag valor numérico

service	Number	service	Number
http	0	link	17
smtp	1	remote job	18
finger	2	gopher	19
domain u	3	name	20
auth	4	whois	21
telnet	5	domain	22
ftp	6	login	23
eco i	7	imap4	24
ntp u	8	ctf	25
ecr i	9	daytime	26
other	10	nntp	27
private	11	uucp	28
pop 3	12	hostnames	29
ftp data	13	vmnet	30
rje	14	pm dump	31
time	15	s9	32
mtp	16		

Tabla 3.4: Service valor numérico

3.2.3. Categorías principales de ciberataques del dataset

El dataset contiene 23 ciberataques distintos, al tener que reducir la carga de datos finalmente el **estudio contempla 18 de ellos** y los logs normales.

Todos ellos están divididos en **4 categorías principales: DOS, probe, R2L y U2R.** [8]

DOS

Los ciberataques **DOS** (Denial of Service) tienen como objetivo que un **servicio o recurso sea inaccesible** para usuarios legítimos. Una de sus caracterísitcas principales es que utilizan la familia de protocolos TCP/IP.

Un ataque **DOS** se puede llevar acabo de la siguientes maneras:

- **Consumiendo recursos operacionales** como el ancho de banda o el tiempo del procesador.
- **Alterando la información de estado**, tales como interrupción de sesiones TCP.
- **Obstrucción de los medios de comunicación** entre los usuarios de un servicio y la víctima del ataque.

En el siguiente capítulo se analizan en profundidad los siguientes ciberataques relacionados con la categoría **DOS: back,land, neptune, pod, smurf y teardrop.**

Probe

Los ciberataques de tipo **probing** buscan **encontrar alguna vulnerabilidad** dentro de la red o de los puertos del objetivo.

La gran mayoría de ciberataques comienzan con este tipo, ya que permiten encontrar debilidades en el objetivo que aprovechar en las siguientes fases del ciberataque.

En este documento comentaremos los siguientes: **ipsweep, nmap, portsweep y satan.**

R2L

R2L es un grupo de ciberataques que tienen como **objetivo exponer las vulnerabilidades y explotar los recursos** de un objetivo al que el **atacante no tiene acceso directo**.

Los ataques de este tipo que se estudian en el siguiente capítulo son: **ftp write, guess password, imap, phf.**

U2R

Los denominados ataques **U2R** se realizan cuando el hacker ya tiene acceso a la máquina objetivo y normalmente tienen como fin **conseguir privilegios sobre el sistema.**

Referentes a esta categoría, en este trabajo están los siguientes ataques: **buffer overflow, load module, perl y rootkit.**

4

Experimentación

4.1. Objetivo de la experimentación

La experimentación está dividida de dos partes, primero se ha realizado un **estudio de los ciberataques del dataset.** El estudio de cada ciberataque incluye su descripción, una correlación entre este y los datos obtenidos en el estudio y una solución.

La segunda parte de la experimentación se centra en la **predicción de futuros ciberataques y la generación de una respuesta rápida.** En el caso actual se incluye tan solo un **playbook manual** ya que no se dispone de los medios para realizar dichas operaciones.

4.2. Estudio del dataset

Si comprender los datos es el primer paso en todo proyecto de ciencia de datos, el segundo es conocer sobre la **materia de los datos**, en este caso, sobre **ciberseguridad** y los ciberataques del dataset.

En la siguiente sección los **ciberataques** se encuentran divididos por las **4 categorías principales** nombradas en el capítulo anterior junto al número de logs de dicho ataque que se incluyen en el estudio.

4.2.1. Categorías y tipos de ciberataques

En esta sección del trabajo se definen los ciberataques del estudio y se relacionan dichas definiciones con las gráficas obtenidas a través del script desarrollado en Python. [9]

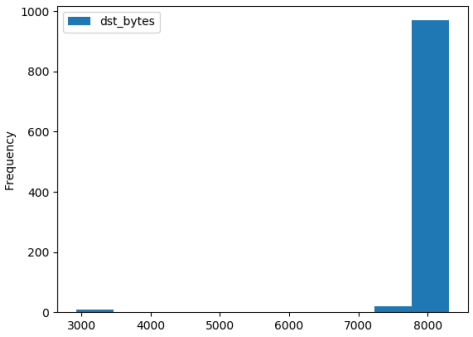
DOS

Ataque 1 ***Back. (1000 logs).*** *Un ataque back consiste en enviar los propios mensajes del objetivo de vuelta a él no permitiendo la comunicación del servidor con los clientes e inundando el servidor con*

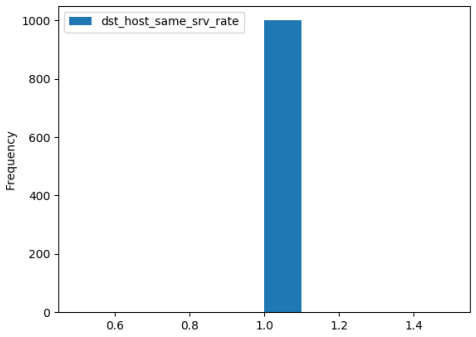
sus propios mensajes.

dst bytes - dst host same srv

En un ataque back, el objetivo recibe de vuelta sus propios mensajes, por tanto es lógico que envíe una gran cantidad de datos y todos sea a través del mismo servicio que es lo que muestran las dos siguientes gráficas.



(a) back - dst bytes

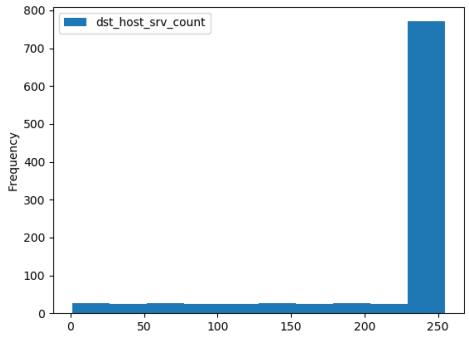


(b) back - dst host same srv

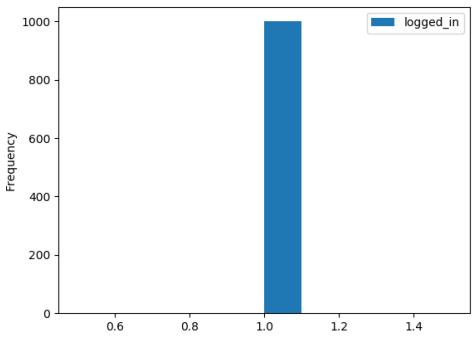
Figura 4.1: back - dst bytes - dst host same srv.

dst host srv count - logged in

Hay una relación directa entre dst bytes y dst host srv count, ya que a mayor cantidad de conexiones más bytes se envían y por supuesto para conseguir que el propio objetivo se inunde necesitas estar estar logueado y tener cierto control de este.



(a) back - dst host srv count histograma



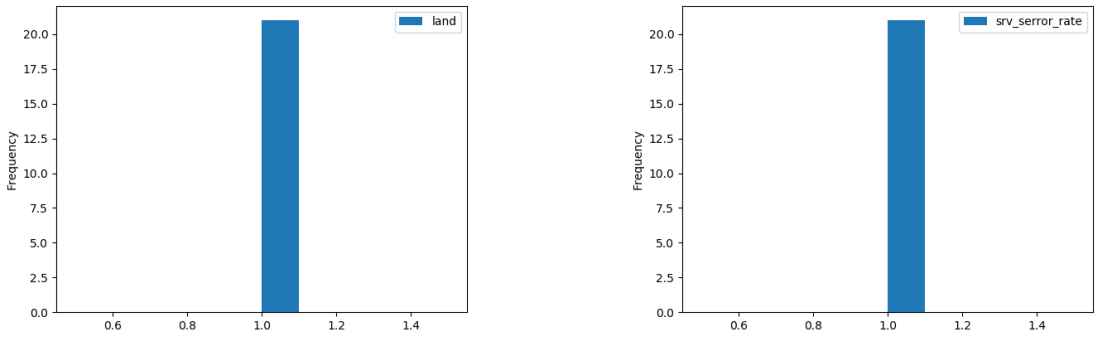
(b) back - logged in histograma

Figura 4.2: back - dst host same - logged in.

Ataque 2 ***Land. (21 logs).*** *Es un tipo de ataque DoS en el que el atacante selecciona como información fuente y destino de un segmento TCP para que sea la misma. Una máquina vulnerable sufrirá caídas o congelaciones debido al procesamiento continuo de dichos segmentos TCP.*

land - srv serror rate

En un ataque de tipo land, la flag del campo land estará levantada y además se producirá un porcentaje alto de errores en el servicio que se esté atacando.



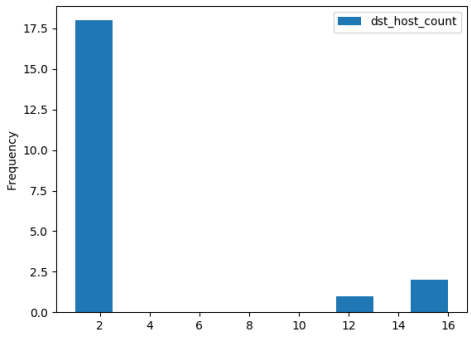
(a) land - land histograma

(b) land - srv serror rate histograma

Figura 4.3: land - land - srv serror rate.

dst host count

Podemos observar como hay una gran varianza en el número de conexiones, esto puede ser debido a la dificultad para bloquear a la máquina objetivo; a mayor cantidad de recursos más conexiones son necesarias.



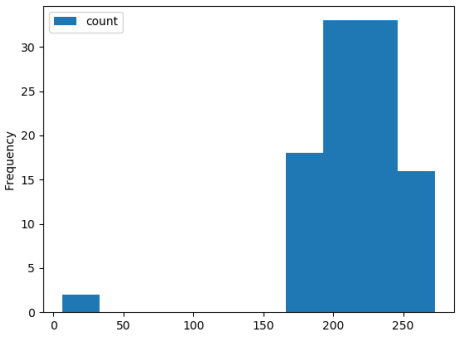
(a) land - dst host count histograma

Figura 4.4: land - dst host count.

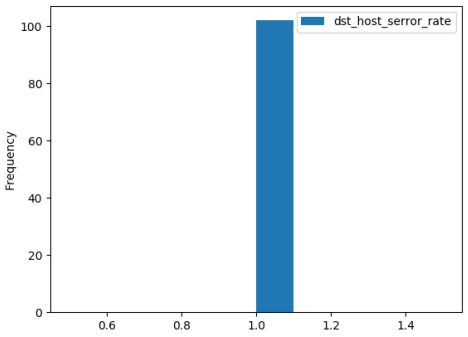
Ataque 3 ***Neptune. (102 logs).*** *Se genera un ataque SYN Flood contra la red objetivo enviando paquetes para establecer una sesión. El host que recibe los paquetes usará sus recursos para esperar la confirmación, lo que hará que no esté disponible para el tráfico legítimo.*

count - dst host serror rate

En un ataque de tipo flood habrá muchas conexiones por tanto el campo count es alto como se muestra en la primera imagen. En el caso de este ataque los errores son de tipo SYN por tanto dst host serror es 1.



(a) neptune - count histograma

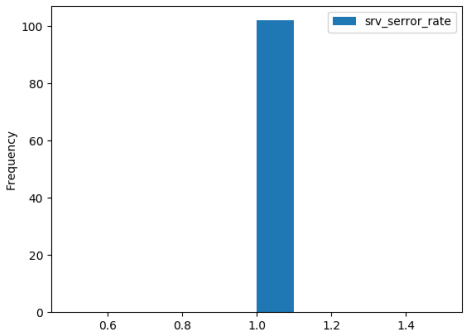


(b) neptune - dst host serror rate histograma

Figura 4.5: neptune - count - dst host serror rate.

srv serror rate

Tal y como pasa con dst host serror la tasa de error de srv serror rate es 1, es decir fallan todos los eventos.



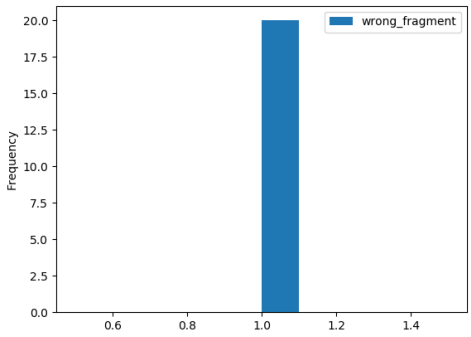
(a) neptune - srv serror rate histograma

Figura 4.6: neptune - count - dst host serror rate.

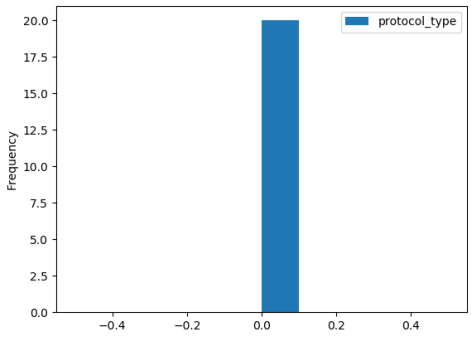
Ataque 4 ***Pod. (20 logs).*** *El ciberataque pod (ping of death) se basa en intentar desestabilizar, congelar o provocar la caída del objetivo enviando paquetes deformados o demasiado grandes usando el comando ping.*

wrong fragment - protocol type

Todos los paquetes están malformados,debido a esto wrong fragment es 1. Ping funciona a través del protocolo ICMP representado por el 0 en protocol type.



(a) pod - wrong fragment histograma



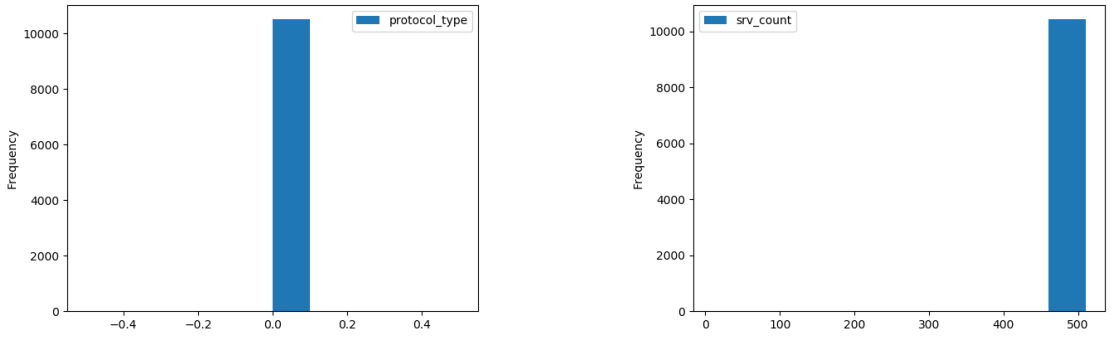
(b) pod - protocol type histograma

Figura 4.7: pod - wrong fragment - protocol type.

Ataque 5 ***Smurf. (10503 logs).*** *Este tipo de ataque DoS se centra en mandar muchos paquetes del protocolo ICMP (Internet Control Message Protocol) con la dirección IP de la victima como source IP a una red usando la IP broadcast de la misma. Al necesitar muchos recursos para gestionar las respuestas de toda la red, el objetivo consume todos los recursos disponibles y deja de ofrecer su servicio a los usuarios.*

protocol type - srv count

Podemos observar como protocol type es 0 que indica que el protocolo es ICMP. Así mismo se produce un gran número de conexiones en el mismo servicio que es lo que bloqueará al objetivo consumiendo sus recursos tal y como se muestra en srv count.



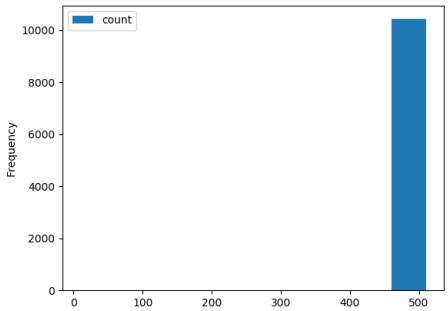
(a) smurf - protocol type histograma

(b) smurf - srv count histograma

Figura 4.8: smurf - protocol type - srv count.

count

count y srv count tienen la misma gráfica ya que todas las conexiones fueron a través del mismo servicio.



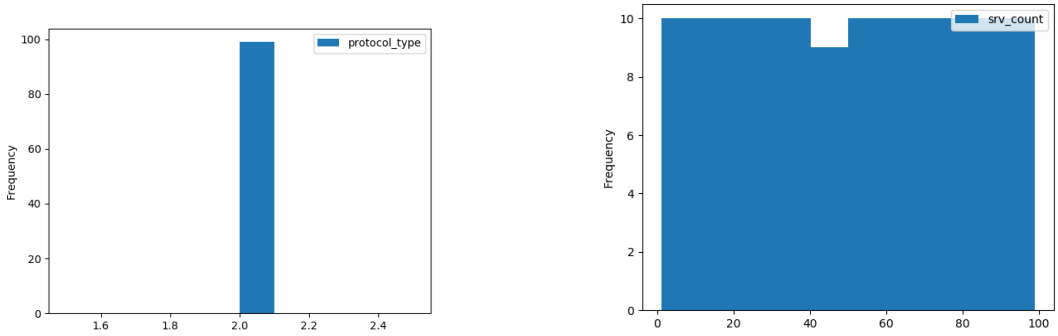
(a) smurf - count histograma

Figura 4.9: smurf - count.

Ataque 6 ***Teardrop. (99 logs).*** *En este tipo de ataque el cliente envía un paquete con información maliciosa de manera que al recomponer el parquete se aproveche de alguna vulnerabilidad.*

protocol type - srv count

El tipo de protocolo de este ataque es UDP. El campo srv count muestra que el número de conexiones varía mucho, esto es debido a la cantidad de paquetes que manda, también varía el número de conexiones para intentar que el paquete malicioso se forme correctamente y pueda atravesar las medidas defensivas.



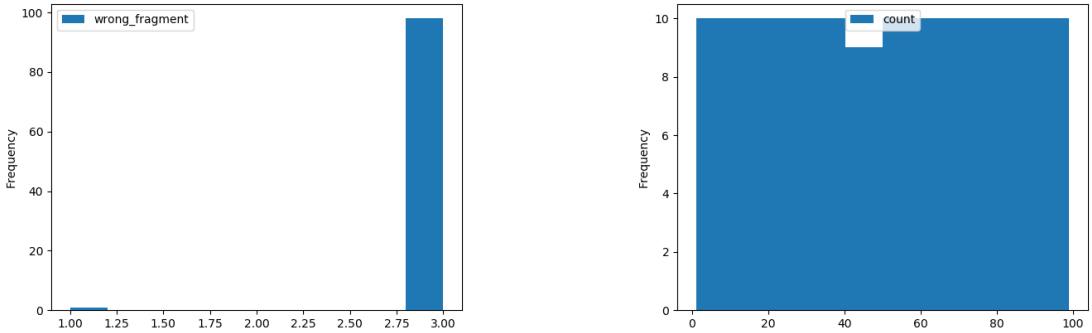
(a) teardrop - protocol type histograma

(b) teardrop - srv count histograma

Figura 4.10: teardrop - protocol type - srv count.

wrong fragment - count

Wrong fragment muestra que hay una media cercana a 3 fragmentos erróneos por ataque y count mantiene la relación con srv count demostrando que las conexiones son siempre a través del mismo servicio.



(a) teardrop - wrong fragment histograma

(b) teardrop - count histograma

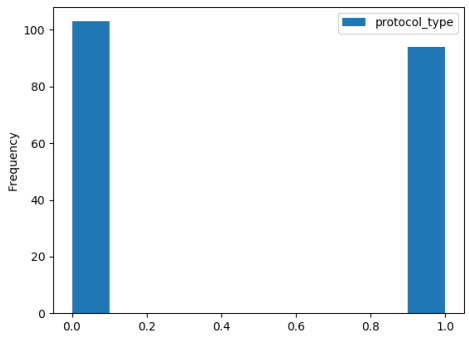
Figura 4.11: teardrop - wrong fragment - count.

Probe

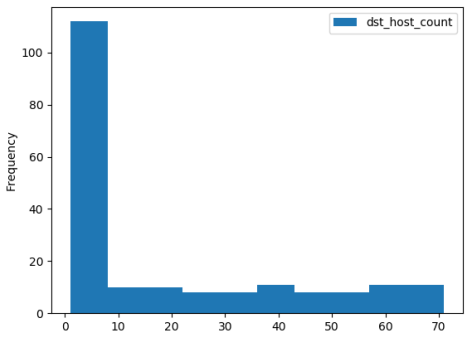
Ataque 7 ***Ipsweep. (197 logs).*** *Un ataque ipsweep ocurre cuando un atacante envía echos ICMP (pings) a varios destinos para que el host responda y revela la dirección ip del objetivo al atacante.*

protocol type - dst host count

Es interesante que el protocol type de ciertos ataques de ipsweep sean TCP cuando debería ser integramente ICMP, quizá se están incluyendo ataques utilizando TCP SYN y por tanto categorizando ataques TCP Sweep como ipsweep. El campo dst host count indica que los ataques tiene bastante diferencia en el número de pings utilizados.



(a) ipsweep - protocol type histograma



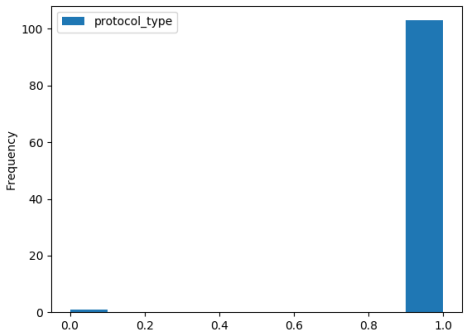
(b) ipsweep - dst host count histograma

Figura 4.12: ipsweep - protocol type - dst host count.

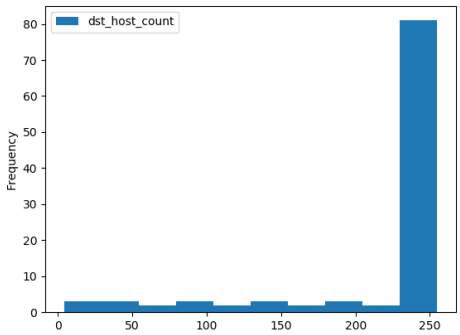
Ataque 8 ***Nmap. (104 logs).*** *Nmap es una herramienta de open source utilizada como escáner de vulnerabilidades y descubrimiento de redes. A la hora de atacar a un objetivo, nmap enumera los puertos abiertos del objetivo y da información útil sobre los servicios de dichos puertos.*

protocol type - dst host count

Nmap trabaja bajo el protocol TCP y podemos observar como realiza un gran número de conexiones en el histograma de dst host count.



(a) nmap - protocol type histograma

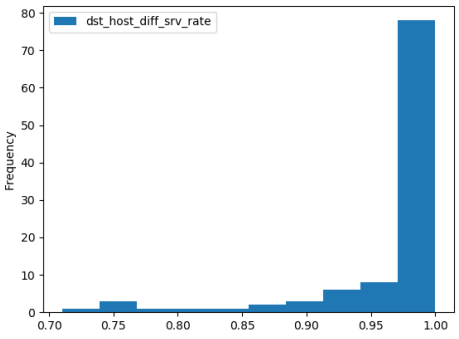


(b) nmap - dst host count histograma

Figura 4.13: nmap - protocol type - dst host count.

dst host diff srv rate

Cada conexión utilizando nmap suele ser en un puerto distinto salvo que estés atacando a varios objetivos en el mismo ataque. Por ello el porcentaje de distintos servicios que se atacan es tan alto en cada ataque.



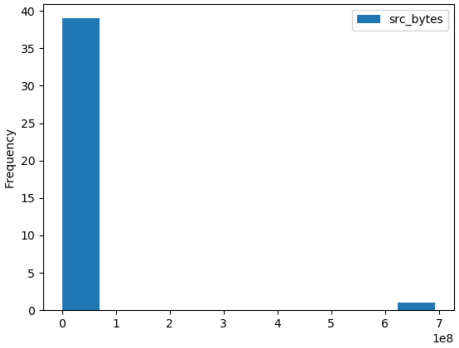
(a) nmap - dst host diff srv rate histograma

Figura 4.14: nmap - dst host diff srv rate.

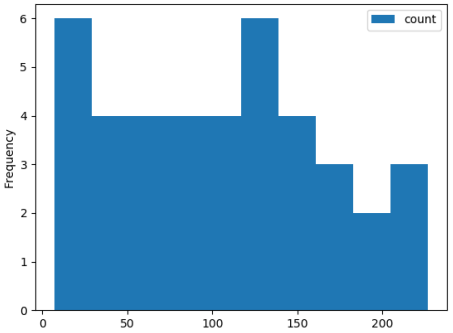
Ataque 9 ***Portsweep. (40 logs).*** *El ataque portsweep se centa en buscar la debilidad en un solo puerto/servicio en una gran cantidad de objetivos y es especialmente usado en los puertos 80 y 443.*

src bytes - count

La gráfica de src bytes muestra que por regla general se suelen enviar pocos bytes en estos ataques, pero hay uno que envía cerca de 7 mega bytes, esto puede ser debido a que se tardó en identificar un puerto/servicio que tuviera una vulnerabilidad. Podemos observar en el histograma de count, que el número de peticiones varías bastante, probablemente el atacante para cuando encuentra una máquina débil al exploit que tiene preparado.



(a) portsweep - src bytes histograma



(b) portsweep - count histograma

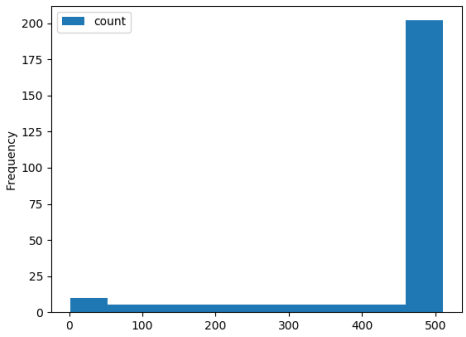
Figura 4.15: portsweep - src bytes - count.

Ataque 10 ***Satan. (252 logs).*** *Satan (Security Administrator Tool for Analyzing Networks), forma parte de un framework de ataque que intenta localizar las vulnerabilidades en las confguraciones de red. También puede identificar los servicios que están actualmente utilizando los puertos.*

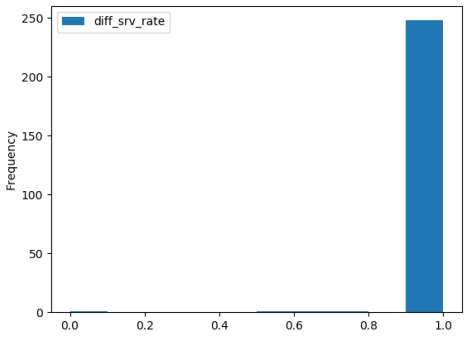
count - diff srv rate

Satán tiene un número grande de conexiones como se puede observar en count, esto, como con el resto de ataques de tipo probing, es debido a que sondea al objetivo en busca de las vulnerabilidades.

El histograma de diff srv rate nos indica que escanea puertos/servicios distintos a la hora de intentar encontrar un fallo en la seguridad.



(a) satan - count histograma



(b) satan - diff srv rate histograma

Figura 4.16: satan - count - diff srv rate.

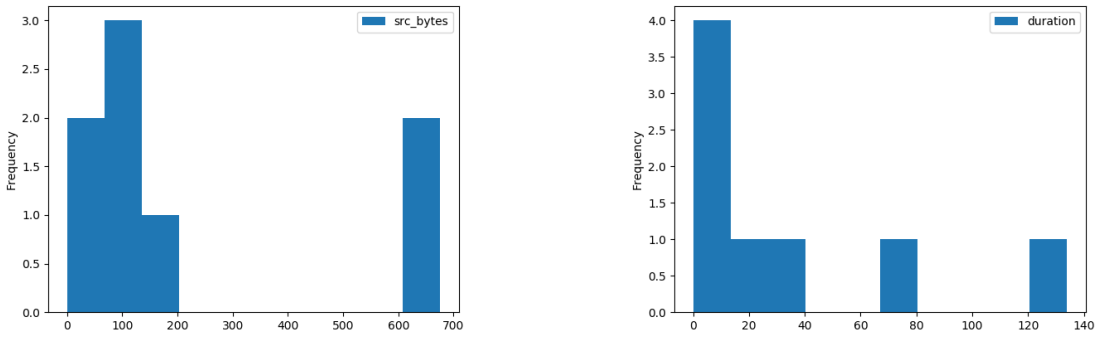
R2L

Ataque 11 ***Ftp write. (8 logs).*** *[10] El atacante es capaz de cargar un fichero malicioso en la máquina objetivo.*

src bytes - duration

El campo src bytes muestra la cantidad de bytes que se envía al destino. Estos bytes son los ficheros que el atacante intenta utilizar para ganar privilegios en la máquina objetivo.

La duración varía bastante en función del fichero a subir y la velocidad de la conexión entre el atacante y el objetivo.



(a) ftp write - src bytes histograma

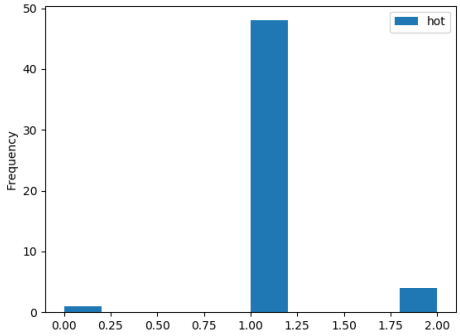
(b) ftp write - duration histograma

Figura 4.17: ftp write - src bytes - duration.

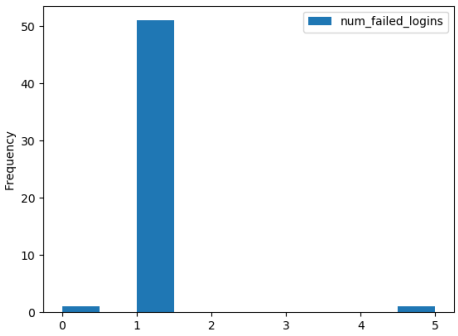
Ataque 12 ***Guess Password. (53 logs).*** *Intentos de adivinar la contraseña y probarla contra un hash criptográfico de la misma.*

hot - num failed logins

Los indicadores hot son unos indicadores de prioridad o importancia, en el caso de guess password pueden indicar que la cuenta que están intentado acceder es una cuenta de administración o importante. Num failed logins muestra que se falló una vez por ataque, pero también se observa un cero, lo que implica que el atacante ha conseguido entrar en alguna de las cuentas.



(a) guess password - hot histograma

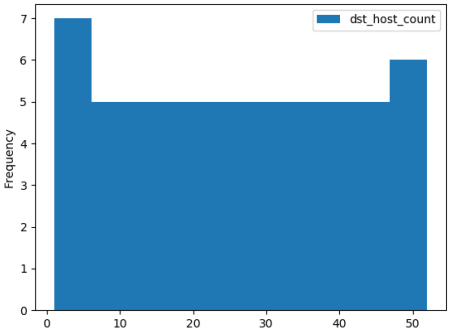


(b) ftp write - num failed logins histograma

Figura 4.18: guess password - hot - num failed logins.

dst host count

La gráfica de dst host count muestra que el número de hosts atacados varía en función del ataque. Esto es debido a que, en función del objetivo, se deben probar distintas cuentas de usuario y la cantidad de hosts en los que se espera a ese usuario puede variar.



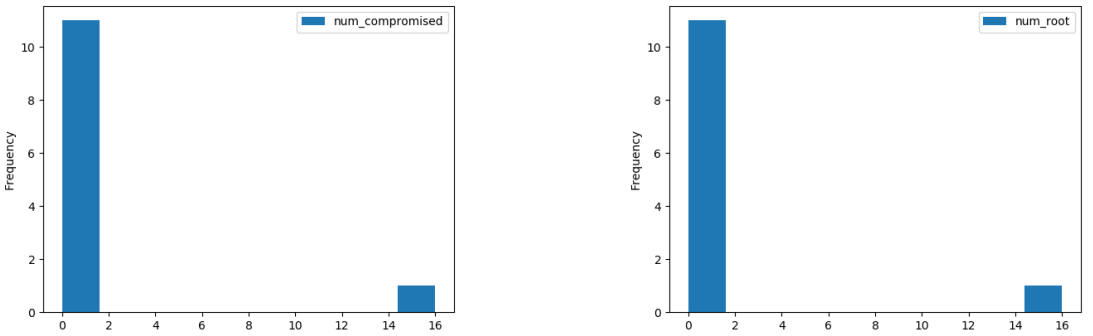
(a) guess password - dst host count histograma

Figura 4.19: guess password - dst host count.

Ataque 13 ***Imap. (12 logs).*** *Intento de autenticación utilizando una misma contraseña junto a un gran número de usuarios.*

num compromised - num root

El número de usuarios comprometidos por ataque suele ser uno como muestra el histograma pero hay un caso dónde se pueden observar 15 usuarios comprometidos, por lo que parece que el atacante ha dado con una contraseña usada para varios usuarios, dichos usuarios son todos root, tal y como indica la imagen de num root.



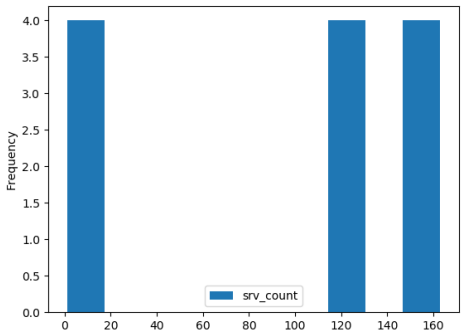
(a) imap - num compromised histograma

(b) imap - num root histograma

Figura 4.20: imap - num compromised - num root.

srv count

El número de peticiones a través de un mismo servicio muestra un patrón de ataque, el atacante ataca 3 veces distintas con un número distintos de usuarios.



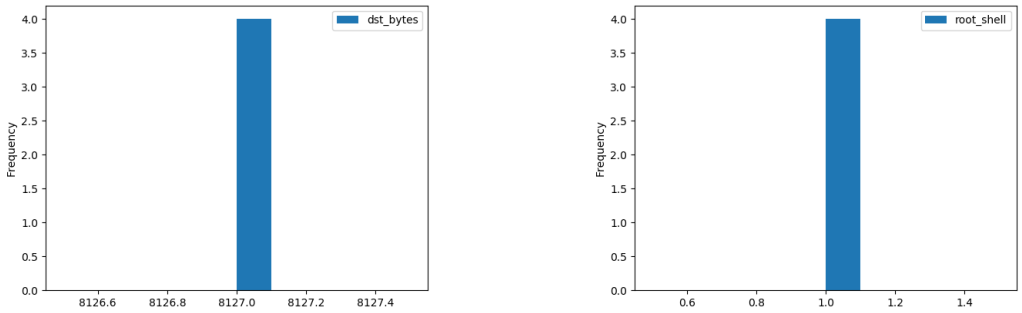
(a) imap - srv count histograma

Figura 4.21: imap - srv count.

Ataque 14 ***Phf. (4 logs).*** *El script phf que está instalado en el directorio cgi-bin por defecto funciona como una agenda de teléfono de las webs a las que has accedido. Los ataques phf se basan en utilizar este fichero de tal manera que aprovechándose de un bug que ocurre cuando la función se usa la función escape shell cmd intenta escapar un carácter especial. Este bug permite al atacante crear una url para conseguir información del fichero, como por ejemplo las contraseñas.*

dst bytes - root shell

Dst bytes es un campo importante ya que muestra que la máquina objetivo está enviando datos al atacante. Root shell indica que el atacante ha conseguido acceso como administrador.



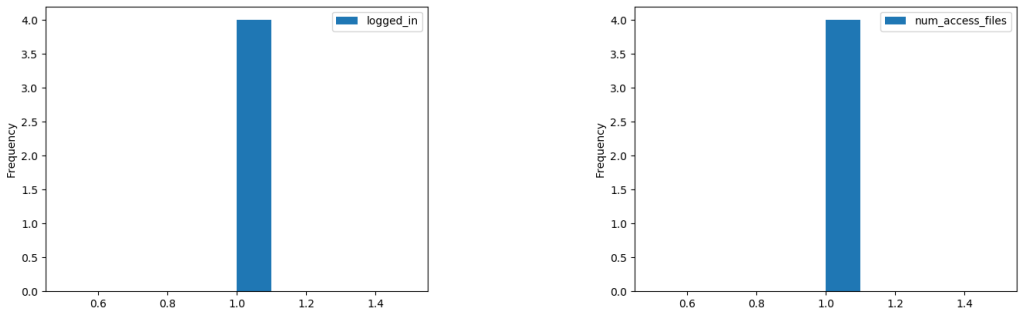
(a) phf - dst bytes histograma

(b) phf - root shell histograma

Figura 4.22: phf - dst bytes - root shell.

logged in - num access files

Al conseguir acceso como administrador, el atacante hace login en la máquina por ello el campo logged in es 1. También podemos observar como el número de ficheros accedidos es 1, es decir han accedido al script phf instalado en cgi-bin.



(a) phf - logged in histograma

(b) phf - num access files histograma

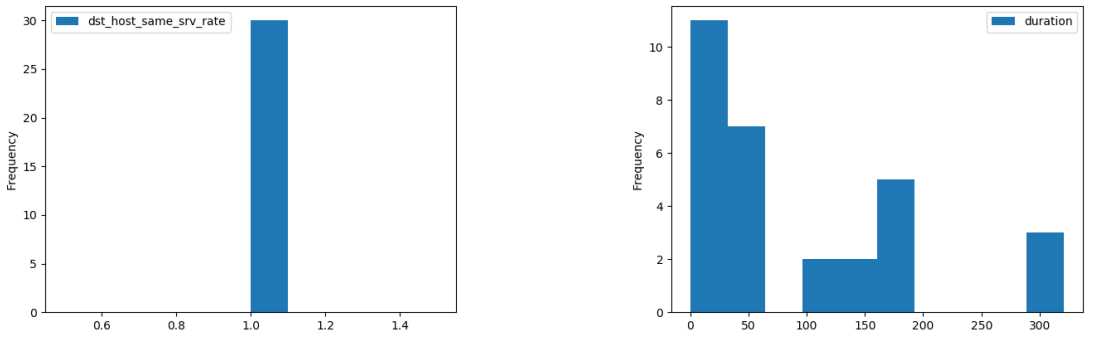
Figura 4.23: phf - logged in - num access files.

U2R

Ataque 15 ***Buffer overflow. (30 logs).*** *[10] Un ataque de buffer overflow intenta utilizar un bug que como su propio nombre indica se basa en sobrescribir memoria adyacente de un programa que no debería modificarse con la finalidad de ganar accesos y privilegios.*

dst host same srv rate - duration

Buffer overflow utiliza el mismo servicio para operar siempre tal y como se muestra en dst host same srv rate y la duración del ataque varía notablemente.



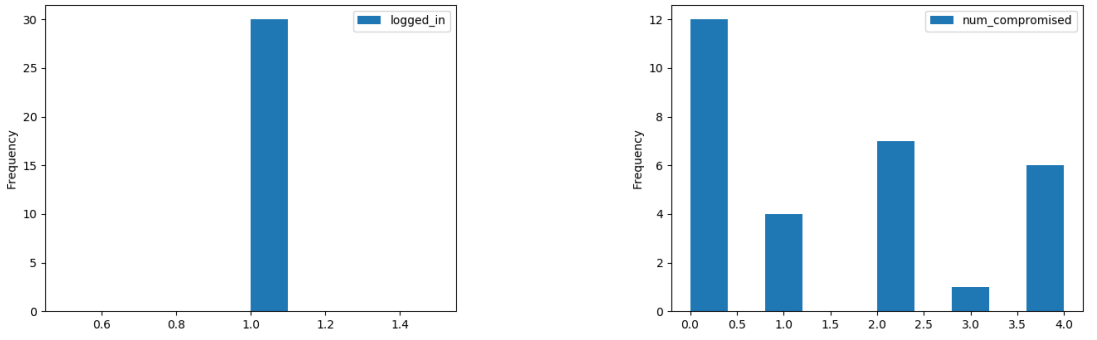
(a) buffer overflow - dst host same srv rate histograma

(b) buffer overflow - duration histograma

Figura 4.24: buffer overflow - dst host same srv rate - duration.

logged in - num compromised

El atacante está logueado en el sistema y pero no siempre consigue comprometer al equipo/usuario tal y como muestra el histograma de num compromised.



(a) buffer overflow - logged in histograma

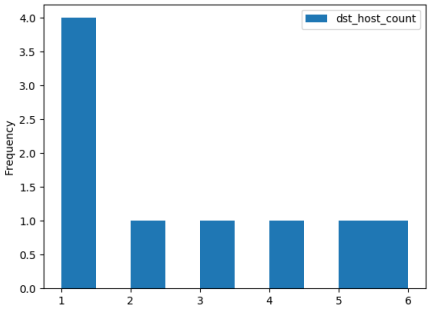
(b) buffer overflow - num compromised histograma

Figura 4.25: buffer overflow - logged in - num compromised.

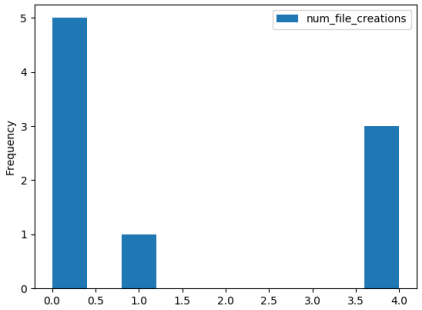
Ataque 16 ***Load Module. (9 logs).*** *Este ataque consiste en utilizar una ventana del sistema del servidor para cargar dos drivers en el sistema y crear un programa especial para utilizar dichos módulos. Debido a un bug del programa un usuario puede conseguir permisos de administrador en dicha máquina.*

dst host count - num file creations

El primer histograma nos indica que no siempre hay las mismas peticiones, es decir que el programa ha sido utilizado de manera distinta, el número de ficheros creados sugiere lo mismo.



(a) load module - dst host count histograma

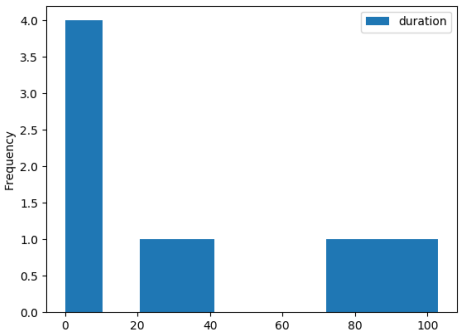


(b) load module - num file creations histograma

Figura 4.26: load module - dst host count - num file creations.

duration

La duración del ataque puede durar hasta 100 segundos según la gráfica de duration, sería necesario tener una mayor cantidad de eventos de load module para conocer mejor las características del ataque. Tanta variabilidad y tan pocos datos hacen que load module sea un ataque difícil de predecir para nuestro sistema actual.



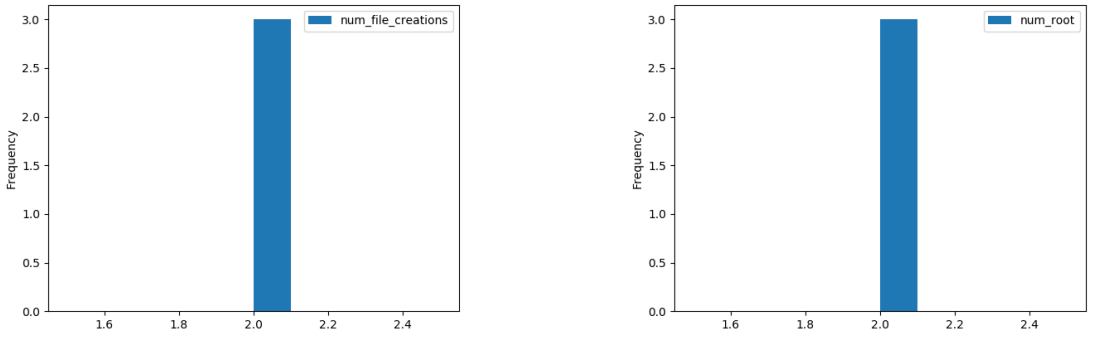
(a) load module - duration histograma

Figura 4.27: load module - duration.

Ataque 17 ***Perl. (3 logs).*** *Los ataques Perl intentan poner el user id del atacante como root en un script e introducir una shell desde la que lanzar comandos.*

num file creations - num root

Num file creations es importante porque el atacante crea el script que le permite identificarse como root tal y como se ve en num root.



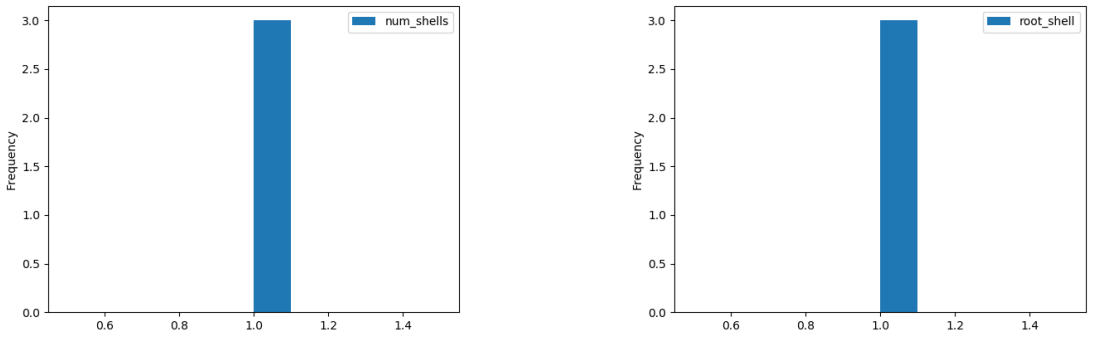
(a) perl - num file creations histograma

(b) perl - num root histograma

Figura 4.28: perl - num file creations - num root.

num shells - root shell

Num shells indica que se crea una shell y que el atacante consigue ser root como se puede observar en las dos imágenes inferiores.



(a) perl - num shells histograma

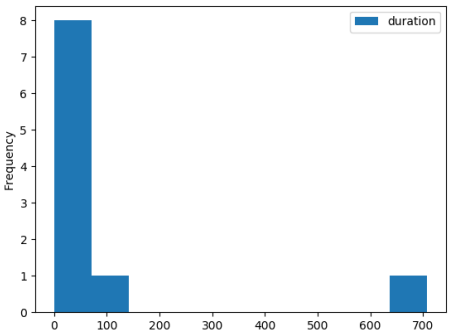
(b) perl - root shell histograma

Figura 4.29: perl - num shells - root shell.

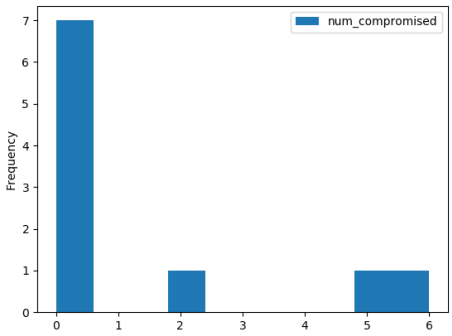
Ataque 18 ***Rootkit. (10 logs).*** *Un rootkit es un tipo de software malicioso que permite a un usuario conseguir privilegios de administrador y alcance a areas restringidas del software. Suele contener un conjunto de herramientas como keyloggers, antivirus disablers, bots or password stealers.*

duration - num compromised

La duración de los ataques rootkit es muy variable ya que depende de la herramienta que desees utilizar. Al tener varias herramientas el número de activos comprometidos puede ser variable. Esto es visible en el segundo histograma.



(a) rootkit - duration histograma

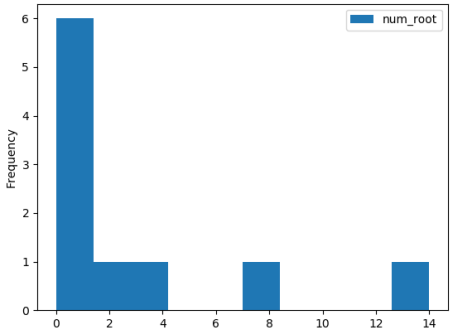


(b) rootkit - num compromised histograma

Figura 4.30: rootkit - duration - num compromised.

num root

El número de accesos root durante el ataque varía desde 0 a 14 accesos. Toda esta variabilidad y el número tan bajo de eventos hacen que el ataque rootkit sea muy difícil de clasificar para los modelos como veremos en el siguiente capítulo.



(a) rootkit - num root histograma

Figura 4.31: rootkit - num root.

4.3. Respuesta a los ataques

Ahora que ya conocemos los ataques del dataset estaríamos interesados en realizar una **respuesta** cuando ocurran. Al no disponer de las herramientas necesarias para llevar esto acabo (no disponemos de una red, ni un listado de máquinas en dicha red, ni de los medios para modificar sus permisos) se ha decidido crear unos **playbooks manuales**.

Los **playbooks** o respuestas a los ataques pueden encontrarse en las siguientes tablas con los ataques ordenados alfabeticamente.

Ataque	Respuesta/ Mitigación
back	Delimitar la cantidad de tráfico que puede llegar a tu asset, de tal manera que no se consuman plenamente los recursos. Esto puede llevarse acabo através de firewalls y WAFs (Web Application Firewall).
buffer overflow	Añadir el hash del fichero y el nombre a dos blacklist distintas para evitar que vuelva a ser un problema en el futuro.
ftp write	Crear un hash del fichero malicioso y añadirlo a una blacklist de tal manera que no pueda moverse más por la red.
guess password	Bloquear la cuenta que está teniendo muchos intentos de inicio de sesión
imap	Si la máquina desde la que se están haciendo los ataques es interna ponerla en cuarentena y buscar al usuario que se ha logueado en ella para bloquearlo. En el caso de ser una máquina externa añadir la ip a una blacklist. También sería necesario revisar si el inicio de sesión en algún momento ha sido exitoso y bloquear dichos usuarios.
ipsweep	Se puede añadir la IPque realiza el escaneo a una blacklist, pero normalmente ya han obtenido la dirección IP que quieren atacar.
land	Bloquear el tráfico que tenga como origen y destino la misma dirección IP. Esto puede conseguirse con micro segmentación de redes.
load module	Revisar las posibles exfiltraciones de datos fuera de tu red. Añadir el hash del programa y el nombre a dos blacklist distintas para evitar ejecuciones en el futuro.
neptune	Delimitar la cantidad de tráfico de paquetes para iniciar sesión dentro de la red.
nmap	Revisar las máquinas escaneadas y listar los servicios de cada una para intentar evitar el futuro ataque utilizando sus vulnerabilidades.

Tabla 4.1: Respuesta ante ataques I

Ataque	Respuesta/ Mitigación
perl	Bloqueo de los usuarios que hayan podido ser target del ataque. Cambio de contraseña y cuarentena a la máquina. Actualización de Perl para evitar esto en futuros ataques
phf	Revisar las exfiltraciones desde esa máquina y a que destino han ido. Poner dicha máquina en cuarentena y listar operaciones que se hayan realizado después y puedan haber comprometido a otras máquinas de la red.
pod	Añadir la dirección IP del atacante a la blacklist para no permitir que se comunique con el objetivo.
portsweep	Mantener atención a los assets que son vulnerables en el puerto que se ha escaneado
rootkit	Añadir el hash y el nombre del programa a las blacklist de ejecución de programas. Revisar las acciones realizadas por el programa y actuar en consecuencia.
satan	Revisar las máquinas escaneadas y listar los servicios de cada una para intentar evitar el futuro ataque utilizando sus vulnerabilidades.
smurf	Delimitar la cantidad de paquetes de protocolo ICMP que puede enviar una máquina.
teardrop	Añadir a una lista negra la configuración de los paquetes erróneos de tal manera que no tengan la disponibilidad de volver a viajar por tu red.

Tabla 4.2: Respuesta ante ataques II

4.4. Modelos para la generación de la predicción

4.4.1. Regresión logística

La **regresión logística** es un método de análisis que se centra en categorizar una variable en función de otras variables independientes o predictoras.

Este método analiza los datos distribuidos de **manera binomial**

$$Y_i \sim B(p_i, n_i)$$

Donde p_i son las probabilidades de éxito o de que ocurra el suceso y con n_i siendo un **ensayo de Bernoulli**, es decir, un expermiento aleatorio en el que solo se pueden obtener dos resultados (éxito o fracaso).

Este método es similar a una **red neuronal** de una sola capa o un solo **perceptrón**:

$$y = \frac{1}{1 + e^{-f(X)}}$$

siendo f(X) una función analítica en X y X el conjunto de todas los sucesos posibles.

4.4.2. Arbol de decisión

Los **árboles de decisión** son un modelo de predicción en el que dado un conjunto de datos se contruye un diagrama lógico que sirven para generar una respuesta en función de una serie de condiciones. Su nombre se debe a la forma que suelen tener los diagramas, ya que parten de un nodo y en base a la serie de condiciones se van generando nuevos nodos los cuales se relacioan mediante flechas, acciones, que te llevan de un nodo a otro.

Este modelo es ampliamente utilizado en teoría de juegos para representar estados y observar cuales son las posibilidades de ganar de cada jugador o sus mejores jugadas.

4.4.3. Kvecinos

El método de **kvecinos** (KNN - k Nearest Neighbors) es un método que busca las observaciones más cercanas a la que se está tratando de predecir y clasifica el evento basándose en aquellas observaciones que le rodean.

Una gran diferencia respecto a otros métodos como la regresión logística o los árboles de decisión es que este algoritmo **no aprende explícitamente un modelo sino que memoriza las instancias del**

entrenamiento y las usa como una base de datos de conocimiento para la fase de predicción.

4.4.4. Análisis lineal discriminante

El **análisis lineal discriminante** (LDA), es un método de clasificación en el que dos o más grupos son conocidos a priori y nuevas observaciones se clasifican en uno de ellos en función de sus características. LDA funciona haciendo uso del **teorema de Bayes**, con él, estima la probabilidad de que un evento pertenezca a una clase tras evaluar las probabilidades de todas las clases le asigna la que tiene mayor probabilidad.

4.4.5. Naive Bayes gaussiano

El clasificador **Naive Bayes** está fundamentado en el **teorema de Bayes** como el algoritmo **LDA**, pero añade algunas **hipótesis simplificadoras**. El apelativo de **naive (ingenuo)** proviene de estas hipótesis, que asumen que las **variables predictoras son independientes.**

Naive Bayes utiliza el **método de máxima verosimiltud** para la estimación de los parámetros del modelo, y al considerar dichas variables independientes se tiene que

$$f(x_1, x_2, ..., x_n) = f(x_1) \cdot f(x_2) \cdot ... \cdot f(x_n)$$

Además necesita una menor cantidad de datos de entrenamiento ya que tan solo debe calcular las varianzas entre las distintas variables predictoras ya que al considerarlas independientes las covarianzas entre ellas son 0.

5

Resultados de la experimentación

En este capítulo revisamos los **resultados de la experimentación**. Primero revisaremos los resultados que arrojan los modelos, en particular su tasa de acierto. Tras ello tomaremos los 3 mejores modelos y los utilizaremos para la generación del **módelo híbrido**.

5.1. Resultados de los modelos

El siguiente diagrama de cajas muestra el **porcentaje de éxito** y la **desviación** de cada uno de los modelos del capítulo anterior:

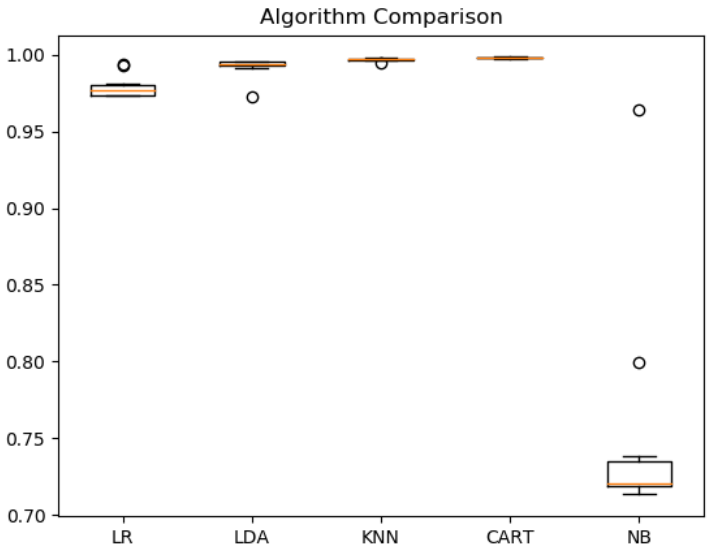


Figura 5.1: Diagrama de caja de los modelos.

Como se puede observar **4 de los modelos** aproximan con más de un **95 %** siendo Naive Bayes el único que no cumple esto. Esto se debe a las **condiciones especiales** de Naive Bayes. Los campos de este dataset por supuesto que tienen relación unos con otros, no son variables independientes. LDA también utiliza el teorema de Bayes y **aproxima mucho mejor.**

La siguiente tabla muestra los resultados exactos del **porcentaje de éxitos** y la **desviación típica** para cada uno de los modelos.

Modelo	Porcentaje de éxito	Desviación típica
Regresión logística	97,92 %	0.7 %
Análisis discriminante lineal	99.20 %	0.7 %
Kvecinos	99.69 %	0.08 %
Árbol de decisión	99.82 %	0.06 %
Naive Bayes	75.34 %	7.42 %

Tabla 5.1: Resultado de los modelos

Por tanto para la **generación del modelo híbrido** se toman los modelos **LDA, kvecinos y el árbol de decisión.**

Antes de avanzar a la generación del modelo híbrido, se realizó un **estudio** del acierto de estos 3 modelos con respecto a **cada uno de los ataques.**

La información se encuentra en 3 columnas:

- **precision**: Es la habilidad del clasificador o modelo para no tomar un evento que no es de un tipo de ataque como ese tipo de ataque.
- **recall**: Es la habilidad del clasificador o modelo para acertar la predicción de todos los eventos relacionados con un tipo de ataque.
- **F1-score**: Es la **media armónica** de las dos columnas anteriores, siendo la media armónica:

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

  Siendo lo ideal, que la **media armónica fuera 1.**
- **support**: Número de eventos de dicho ciberataque en la muestra. [11]

Los datos del modelo **árbol de decisión** son

Cyberattack	precision	recall	f1-score	support
back.	1.00	1.00	1.00	287
buffer overflow.	0.75	0.50	0.60	12
ftp write.	1.00	0.50	0.67	2
guess passwd.	1.00	0.88	0.94	17
imap.	1.00	1.00	1.00	2
ipsweep.	1.00	0.97	0.99	78
land.	0.50	0.25	0.33	4
loadmodule.	0.00	0.00	0.00	4
neptune.	0.97	1.00	0.99	36
nmap.	1.00	0.96	0.98	28
normal.	1.00	1.00	1.00	11360
pod.	1.00	1.00	1.00	6
portsweep.	0.83	1.00	0.91	5
rootkit.	0.00	0.00	0.00	1
satan.	0.99	0.99	0.99	90
smurf.	1.00	1.00	1.00	3161
teardrop.	1.00	1.00	1.00	37

Tabla 5.2: Árbol - Reporte de clasificación

Los valores bajos se corresponden con valores de aquellos ataques en los que se dispone de un **support bajo** y de **mucha variabilidad**, por lo que es de esperar que a mayor afluencia de datos se comiencen a detectar mejor.

El modelo **kvecinos** tiene la siguiente tabla de datos

Cyberattack	precision	recall	f1-score	support
back.	1.00	0.99	0.99	287
buffer overflow.	1.00	0.50	0.67	12
ftp write.	0.00	0.00	0.00	2
guess passwd.	1.00	0.94	0.97	17
imap.	1.00	1.00	1.00	2
ipsweep.	0.92	0.87	0.89	78
land.	1.00	0.50	0.67	4
loadmodule.	0.00	0.00	0.00	4
neptune.	0.86	1.00	0.92	36
nmap.	0.93	0.89	0.91	28
normal.	1.00	1.00	1.00	11360
pod.	1.00	1.00	1.00	6
portsweep.	0.43	0.60	0.50	5
rootkit.	0.00	0.00	0.00	1
satan.	1.00	0.88	0.93	90
smurf.	1.00	1.00	1.00	3161
teardrop.	1.00	1.00	1.00	37

Tabla 5.3: Kvecinos - Reporte de clasificación

Kvecinos tiene **mayor eficacia** a la hora de predecir los ataques de tipo **buffer overflow y land** respecto al árbol de decisión pero **pierde eficacia** en los ataques **ftp write (no acierta ninguno), ipsweep, neptune y portsweep.**

Por último para el **modelo LDA** se tiene la siguiente tabla de datos

Cyberattack	precision	recall	f1-score	support
back.	0.96	1.00	0.98	287
buffer overflow.	0.86	0.50	0.63	12
ftp write.	1.00	0.50	0.67	2
guess passwd.	1.00	1.00	1.00	17
imap.	0.67	1.00	0.80	2
ipsweep.	0.87	0.86	0.86	78
land.	0.50	0.50	0.50	4
loadmodule.	0.00	0.00	0.00	4
neptune.	0.97	1.00	0.99	36
nmap.	1.00	1.00	1.00	28
normal.	1.00	1.00	1.00	11360
pod.	1.00	1.00	1.00	6
portsweep.	0.50	1.00	0.67	5
rootkit.	0.00	0.00	0.00	1
satan.	1.00	0.93	0.97	90
smurf.	1.00	1.00	1.00	3161
teardrop.	1.00	1.00	1.00	37

Tabla 5.4: LDA - Reporte de clasificación

El modelo de análisis discriminante lineal posee una **mayor tasa de acierto** para el ciberataque **buffer overflow** respecto al modelo de árbol de decisión pero tiene **menor tasa acierto** para los ciberataques **back, imap, ipsweep y portsweep.**

5.2. Generación del modelo híbrido

El algoritmo del **modelo híbrido** utiliza las **predicciones de los 3 modelos anteriores** para generar su propia predicción de la siguiente manera.

- Si las **3 predicciones** coinciden toma el valor de cualquiera de ellas.
- Si la **predicción del mejor modelo predictivo coincide con el segundo o el tercero** entonces la predicción toma el valor del **mejor modelo predictivo.**
- Si la **predicción del segundo modelo predictivo y del tercero son iguales** y distintas al del primero, la predicción toma el **valor del segundo modelo predictivo.**
- En el caso de que todas las predicciones sean distintas se toma el valor del **mejor modelo predictivo.**

El porcentaje de éxito para este modelo predictivo es del **99.82 %** similar al del árbol de decisión. Esto no quiere decir que ambos modelos sean iguales.

Cyberattack	precision	recall	f1-score	support
back.	1.00	1.00	1.00	287
buffer overflow.	0.86	0.50	0.63	12
ftp write.	1.00	0.50	0.67	2
guess passwd.	1.00	0.94	0.97	17
imap.	1.00	1.00	1.00	2
ipsweep.	1.00	0.96	0.98	78
land.	0.50	0.25	0.33	4
loadmodule.	0.00	0.00	0.00	4
neptune.	0.97	1.00	0.99	36
nmap.	1.00	0.96	0.98	28
normal.	1.00	1.00	1.00	11360
pod.	1.00	1.00	1.00	6
portsweep.	0.62	1.00	0.77	5
rootkit.	0.00	0.00	0.00	1
satan.	1.00	0.94	0.97	90
smurf.	1.00	1.00	1.00	3161
teardrop.	1.00	1.00	1.00	37

Tabla 5.5: Modelo Híbrido - Reporte de clasificación

Podemos observar revisando ambas tablas como **mejora la detección del ataque buffer overflow pero reduce la detección en portsweep y satán.**

6

Conclusiones y trabajo futuro

6.1. Conclusiones

Se ha conseguido **entender y dar un estudio detallado del dataset KDD Cup 1999** tanto de los ataques como de los campos que lo componen y relacionando los ataques con campos que mejor los definen.

Se ha podido **simular el trabajo de predicción de amenazas llevado a cabo por un SIEM** pero utilizando modelos de aprendizaje con una tasa de éxito superior al **99 %.**

Se han incluido una serie de respuestas a dichos ataques en forma de **playbooks manuales**, que con los medios adecuados, podrían funcionar de manera automática simulando el funcionamiento de un SOAR.

6.2. Trabajo futuro

Este TFG podría ampliarse de acuerdo a los siguientes puntos

- Encontrar un **modelo híbrido mejor** utilizando una máquina más potente que permita un input de datos mayor.
- Realizar un estudio de los ciberataques de este dataset junto a la **matriz de tácticas y técnicas de Mitre** para conocer que tácticas y técnicas se podrían cubrir con la protección necesaria para estos ciberataques.
- Hallar información sobre la **evolución de estos ciberataques** hasta nuestros días, recordemos que el dataset es de la KDD Cup 1999 y ya han pasado **21 años.**
- Realizar un trabajo similar a este con un **dataset más moderno** para comprobar si el modelo híbrido sigue siendo una buena solución.

Bibliografía

[1] I. I. N. de Ciberseguridad. INCIBE - Etapas de la Kill Chain [fecha de último acceso 06/07/2020].

[2] S. Engelbrecht. Evolution of SOAR [fecha de último acceso 06/07/2020].

[3] S. Phantom. Splunk phantom-SOAR[fecha de último acceso 02/07/2020].

[4] E. G. Anna L. Buczak, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE COMMUNICATIONS SURVEYS TUTORIALS*, vol. 1, no. 1, pp. 1–24, 2016. .

[5] S. R. Natarajan Meghanathan, Dhinaharan Nagamalai, *Networks and Communications (NetCom2013)*. Springer, 2014.

[6] I. K. K. H. Jonghoon Lee, Jonghyun Kim, "Cyber threat detection based on artificial neural networks using event profiles," *IEEE Access*, vol. 1, no. 1, pp. 1–20, 2019. Read the article.

[7] U. de Irvine KDD Cup, 1999.

[8] S. Paliwal and R. Gupta, "Denial-of-service, probing remote to user (r2l) attack detection using genetic algorithm," *International Journal of Computer Applications*, vol. 1, no. 1, pp. 1–6, 2012. Read the article.

[9] A. S. Janusz S. Kowalik, Janusz Gorski, *Cyberspace Security and Defense: Research Issues*. Springer, 2006.

[10] A. W. Wenke Lee, Ludovic Me, *Recent Advances in Intrusion Detection*. Springer, 2001.

[11] S. portal oficial. Link: Métricas de sklearn[fecha de último acceso 02/07/2020].

[12] J. Brownlee. Machine Learning Libraries for Python and first steps guide [fecha de último acceso 07/07/2020].

Apéndices

A

Código de la aplicación Python

Se ha dividido el código del script en varias partes en función del objetivo de cada una. [12]

Código A.1: En esta figura podemos ver las librerías utilizadas para el proyecto. Os y sys se usan para crear los ficheros y las carpetas de las imágenes para el TFG. Pandas tiene las funciones necesarias para el manejo del dataset y sklearn tiene los modelos de predicción

```
1  # Libraries for Machine Learning
2  import os
3  import sys
4  from pandas import read_csv
5  import pandas as pd
6  from matplotlib import pyplot
7  from sklearn.model_selection import train_test_split
8  from sklearn.model_selection import cross_val_score
9  from sklearn.model_selection import StratifiedKFold
10 from sklearn.metrics import classification_report
11 from sklearn.metrics import confusion_matrix
12 from sklearn.metrics import accuracy_score
13 from sklearn.linear_model import LogisticRegression
14 from sklearn.tree import DecisionTreeClassifier
15 from sklearn.neighbors import KNeighborsClassifier
16 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
17 from sklearn.naive_bayes import GaussianNB
```

Código A.2: En esta parte de código se puede obersvar la inicialización del datset así como el array de campos y los tipos de ataque bajo el nombre de classes. Short classes se ha utilizado en las gráficas ya que algunos nombres eran especialmente largos. Dataset y Dataset Numbers son el mismo dataset pero modificando los campos de tipo string por otros numéricos para que los modelos pudieran intepretarlos, los cambios están explicados en el capítulo 3.

```
18 fileRoute =
       "C:/Users/Ferpi/Downloads/TFG-Informatica/Dataset/kddcup_data_10_percent/Dataset-TFG-10.csv"
19 fileRouteNumbers =
       "C:/Users/Ferpi/Downloads/TFG-Informatica/Dataset/kddcup_data_10_percent/Dataset-TFG-10-numbers.csv"
20 fields = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment',
       'urgent',
21          'hot', 'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell', 'su_attempted',
                'num_root',
22          'num_file_creations', 'num_shells', 'num_access_files', 'num_outbound_cmds', 'is_host_login',
                'is_guest_login',
23          'count', 'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate',
                'same_srv_rate',
24          'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
                'dst_host_same_srv_rate',
25          'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
                'dst_host_serror_rate',
26          'dst_host_srv_serror_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'class']
27 numericfields = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land',
       'wrong_fragment', 'urgent',
28          'hot', 'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell', 'su_attempted',
                'num_root',
29          'num_file_creations', 'num_shells', 'num_access_files', 'num_outbound_cmds', 'is_host_login',
                'is_guest_login',
30          'count', 'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate',
                'same_srv_rate',
31          'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
                'dst_host_same_srv_rate',
32          'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
                'dst_host_serror_rate',
33          'dst_host_srv_serror_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate']
34 classes = ['back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.', 'ipsweep.', 'land.',
35            'loadmodule.', 'neptune.', 'nmap.', 'normal.', 'perl.', 'phf.', 'pod.', 'portsweep.',
36            'rootkit.', 'satan.', 'smurf.', 'teardrop.']
37 shortclasses = ['back', 'buof', 'ftpw', 'gupas', 'imap', 'ipswe', 'land',
38            'ldmd', 'npt', 'nmap', 'nrml', 'perl', 'phf', 'pod', 'pswe',
39            'rtkt', 'satan', 'smurf', 'trdrp']
40 dataset = read_csv(fileRoute, names=fields)
41 datasetNumbers = read_csv(fileRouteNumbers, names=fields)
```

Código A.3: En esta figura se muestra la creación de dos imágenes para cada ataque-campo una del tipo caja y otra de tipo histograma.

```
for clas in classes:
    clas2 = clas[:-1]
    if not os.path.exists("infoperclass/"+clas2):
        os.mkdir("infoperclass/"+clas2)
    if not os.path.exists("infoperclass/"+clas2+"/FieldGraphics"):
        os.mkdir("infoperclass/"+clas2+"/FieldGraphics")
    if not os.path.exists("infoperclass/"+clas2+"/FieldHistograms"):
        os.mkdir("infoperclass/"+clas2+"/FieldHistograms")
    minidataset = datasetNumbers[datasetNumbers["class"].str.endswith(clas)]
    g = open("infoperclass/"+clas2+"/"+clas2+"-information.txt", "w+")
    information=str(minidataset.info(buf=g))
    g.close()
    f = open("infoperclass/"+clas2+"/"+clas2+"-describe.txt", "w+")
    with pd.option_context('display.max_columns', 50):
        f.write(str(minidataset.describe(include= 'all')))
    f.close()
    print(clas2 + "-Describe_+_Info_generados_en_su_carpeta_correspondiente")
    for field in numericfields:
        columnaDS = minidataset[[field]]
        plot = columnaDS.plot(kind='box')
        pyplot.savefig('infoperclass/'+clas2+'/FieldGraphics/'+clas2+'-'+field+'.png')
        plot = columnaDS.plot(kind='hist')
        pyplot.savefig('infoperclass/' + clas2 + '/FieldHistograms/' + clas2 + '-' + field + '.png')
        print(clas2+"-"+field + "-Plot_generados_en_su_carpeta_correspondiente")
        pyplot.cla()
        pyplot.clf()
        pyplot.close('all')
```

.

Código A.4: En el código de esta figura se muestra como se crean las imágenes generales que se han utilizado para realizar un primer estudio

```
68 if not os.path.exists("infoperclass/GeneralImages"):
69     os.mkdir("infoperclass/GeneralImages")
70 for field in numericfields:
71     a = []
72     for clas in classes:
73         minidataset = datasetNumbers[datasetNumbers["class"].str.endswith(clas)]
74         columnaDS = minidataset[[field]]
75         Value=int(columnaDS.mean())
76         a.append(Value)
77     pyplot.plot(shortclasses, a, 'ro')
78     pyplot.xticks(rotation=90)
79     pyplot.savefig('infoperclass/GeneralImages/'+field+'.png')
80     pyplot.cla()
81     pyplot.clf()
82     pyplot.close('all')
```

Código A.5: El código de esta figura muestra como se divide el dataset en dos partes X e y, y se crea el entrenamiento para los modelos con un 70 % de los datos para el entrenamiento y un 30 % para las predicciones. En la siguiente etapa se evalúa la predicción de cada modelo con la media y la desviación típica de cada uno de ellos y se crea un diagrama de caja dónde se muestran los datos de los resultados de los modelos.

```
83  # Split-out validation dataset
84  array = datasetNumbers.values
85  X = array[:, :41]
86  y = array[:, 41]
87  X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.30, random_state=1)
88
89  # Several different approches by algorithms-models
90  models = []
91  models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
92  models.append(('LDA', LinearDiscriminantAnalysis()))
93  models.append(('KNN', KNeighborsClassifier()))
94  models.append(('CART', DecisionTreeClassifier()))
95  models.append(('NB', GaussianNB()))
96  # evaluate each model in turn
97  results = []
98  names = []
99  for name, model in models:
100     print("I'm with "+name)
101     kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
102     cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
103     results.append(cv_results)
104     names.append(name)
105     print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
106
107 # Compare Algorithms
108 pyplot.boxplot(results, labels=names)
109 pyplot.title('Algorithm Comparison')
110 #pyplot.show()
111 pyplot.savefig('Models/Models.png')
```

Código A.6: En esta figura se ve la creación de los ficheros con información extra para el modelo árbol de decisión.

```
112 # Make predictions on validation dataset
113 arbol = DecisionTreeClassifier()
114 arbol.fit(X_train, Y_train)
115 predictionsArbol = arbol.predict(X_validation)
116 if not os.path.exists("Models/Tree"):
117     os.mkdir("Models/Tree")
118 # Evaluate predictions
119 with open('Models/Tree/accuracy_score.txt', 'a') as file:
120     print(accuracy_score(Y_validation, predictionsArbol), file=file)
121 with open('Models/Tree/confusion_matrix.txt', 'a') as file:
122     print(confusion_matrix(Y_validation, predictionsArbol), file=file)
123 with open('Models/Tree/classification_report.txt', 'a') as file:
124     print(classification_report(Y_validation, predictionsArbol), file=file)
```

Código A.7: En esta figura se ve la creación de los ficheros con información extra para el modelo Kvecinos.

```
125 Kvecinos = KNeighborsClassifier()
126 Kvecinos.fit(X_train, Y_train)
127 predictionsKvecinos = Kvecinos.predict(X_validation)
128 # Evaluate predictions
129 if not os.path.exists("Models/Kneighbours"):
130     os.mkdir("Models/Kneighbours")
131 with open('Models/Kneighbours/accuracy_score.txt', 'a') as file:
132     print(accuracy_score(Y_validation, predictionsKvecinos), file=file)
133 with open('Models/Kneighbours/confusion_matrix.txt', 'a') as file:
134     print(confusion_matrix(Y_validation, predictionsKvecinos), file=file)
135 with open('Models/Kneighbours/classification_report.txt', 'a') as file:
136     print(classification_report(Y_validation, predictionsKvecinos), file=file)
```

Código A.8: En esta figura se ve la creación de los ficheros con información extra para el modelo análisis discriminante lineal.

```
137 LDA = LinearDiscriminantAnalysis()
138 LDA.fit(X_train, Y_train)
139 predictionsLDA = LDA.predict(X_validation)
140 # Evaluate predictions
141 if not os.path.exists("Models/LDA"):
142     os.mkdir("Models/LDA")
143 with open('Models/LDA/accuracy_score.txt', 'a') as file:
144     print(accuracy_score(Y_validation, predictionsLDA), file=file)
145 with open('Models/LDA/confusion_matrix.txt', 'a') as file:
146     print(confusion_matrix(Y_validation, predictionsLDA), file=file)
147 with open('Models/LDA/classification_report.txt', 'a') as file:
148     print(classification_report(Y_validation, predictionsLDA), file=file)
```

Código A.9: El código de esta figura muestra el pequeño algoritmo con el que el modelo híbrido forma su respuesta en función de los tres modelos que mejor porcentaje de éxito tienen en la predicción. También se crean los ficheros de errores que me han permitido comparar los resultados del árbol de decisión con respecto a los del modelo híbrido.

```
149 finalPrediction = []
150 ErroresOriginal=0
151 ErroresCombinado=0
152 #Modelo Híbrido
153 for i in range(len(Y_validation)):
154     if(predictionsArbol[i] == predictionsKvecinos[i] == predictionsLDA[i]):
155         finalPrediction.append(predictionsArbol[i])
156     elif (predictionsArbol[i] == predictionsKvecinos[i]):
157         finalPrediction.append(predictionsArbol[i])
158     elif(predictionsArbol[i] != predictionsKvecinos[i]):
159         if(predictionsArbol[i] == predictionsLDA[i]):
160             finalPrediction.append(predictionsArbol[i])
161         elif (predictionsKvecinos[i] == predictionsLDA[i]):
162             finalPrediction.append(predictionsKvecinos[i])
163         else:
164             finalPrediction.append(predictionsArbol[i])
165
166     if(Y_validation[i] != predictionsArbol[i]):
167         ErroresOriginal=ErroresOriginal+1
168         with open('Models/original_errors.txt', 'a') as file:
169             print(str(ErroresOriginal) + ". Esperado: " + Y_validation[i] + " // Arbol: " +
                    predictionsArbol[i], file=file)
170     if (Y_validation[i] != finalPrediction[i]):
171         ErroresCombinado = ErroresCombinado + 1
172         with open('Models/final_errors.txt', 'a') as file:
173             print(str(ErroresCombinado) + ". Esperado: " + Y_validation[i] + " // Arbol: " +
                    predictionsArbol[
174             i] + " // Kvecinos: " + predictionsKvecinos[i] + " // LDA: " + predictionsLDA[i] + " //
                    Final: " + finalPrediction[i], file=file)
```

Código A.10: En esta figura se ve la creación de los ficheros con información extra para el modelo híbrido.

```
175 if not os.path.exists("Models/Final"):
176     os.mkdir("Models/Final")
177 with open('Models/Final/accuracy_score.txt', 'a') as file:
178     print(accuracy_score(Y_validation, finalPrediction), file=file)
179 with open('Models/Final/confusion_matrix.txt', 'a') as file:
180     print(confusion_matrix(Y_validation, finalPrediction), file=file)
181 with open('Models/Final/classification_report.txt', 'a') as file:
182     print(classification_report(Y_validation, finalPrediction), file=file)
```


UAM
UNIVERSIDAD AUTONOMA
DE MADRID